Pokemon Type Rankings

Trevor Squires

Last Updated: February 4, 2023

1 Introduction

With the release of another new generation of Pokemon comes yet another unique mechanic to the Pokemon Scarlet and Violet games. The generation 9 mechanic terastallization allows for the changing of a single Pokemon's type once per battle. It should come as no surprise to any competitive player that the introduction of terastallization allows for many previously lesser used Pokemon to become viable. Although a Pokemon's viability is often very muddied between many factors such as its move pool, stat distribution, ability, and even sometimes the meta, a Pokemon's type tends to be extremely relevant.

For those new to the games, each Pokemon has either one or two types associated with it. Furthermore, each attack has a corresponding type as well. When an attack of a certain type is launched into a Pokemon there is a damage multiplier applied depending on the type matchup. For example, water has a 2x or "super effective" matchup into fire. Thus, when a water attack is used into a fire type, the damage is doubled. The full type chart is shared in Figure 1.

Much of a Pokemon's viability is tied to its type since it plays such a massive role in damage calculations. A good offensive type means that a Pokemon of that type is assumed to learn many moves that tend to hit much of the rest of the types for a lot of damage. A good defensive type is one that takes only a fraction of the normal damage in most cases. As a result, it seems natural to question how powerful is each typing on its own? In this document, we will explore this very question. In doing so, we will construct a solid starting point for anyone wishing to optimize their Pokemon's tera-type.

Considerations

Determining the "best" type is not quite straight-forward. After all, if one were to suggest that any type, say fire, is the best type, it would be easy to counter by saying that water beats fire both offensively and defensively in a 1v1. Furthermore, there are many indirect effects that a Pokemon's type has on it in addition to type match-ups. For example, while grass Pokemon tend to have many weaknesses, they also benefit from being immune to certain status moves such as spore. The impact of these indirect factors are quite difficult to determine. As such, we should explicitly state our assumptions and keep them in consideration when reviewing our results.

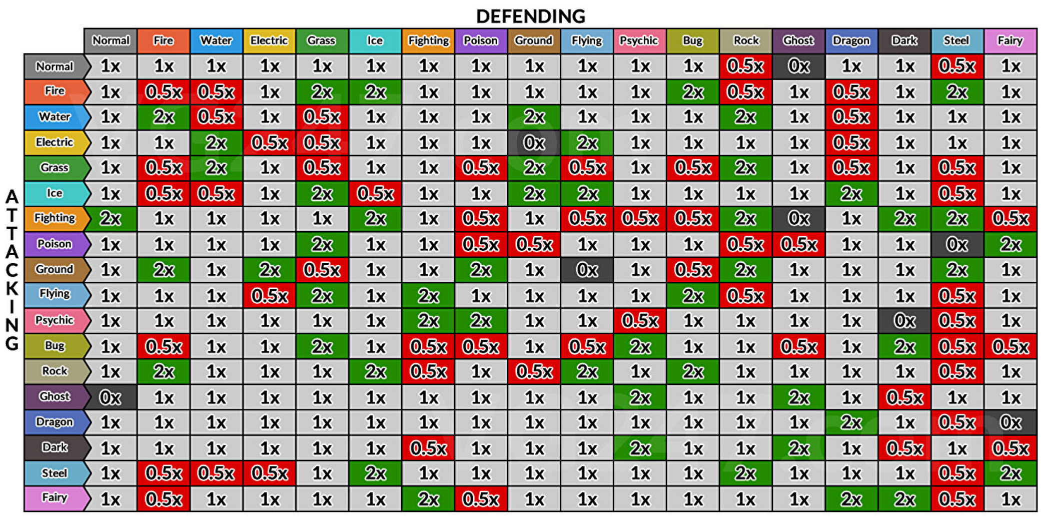
Specifically, we will attempt to rank each type according to its match-ups in a vacuum. We will not be considering the effects of

- additional bonuses such as immunity to prankster, freeze, bonuses in sand, etc.
- the move sets of certain types such as the tendency for water types to learn ice beam or the wide distribution of close combat
- the average base stat total (bst) of each type

Instead, we will only be considering how each type matches up against the other types. We will follow up with some discussion and how these results compare to the standard understanding of typings. In the next two sections, we will propose an approach for ranking all of the types in Pokemon. For the less technically interested, feel free to skip to the last section for the final results.

2 Computing Rankings via Stationary Distribution

At the time of this writing, there are currently 18 different types in Pokemon. However, as mentioned previously, they each have unique advantages and disadvantages over each other. As such, it can be a bit messy when trying



DEFENDING

ATTACKING	Normal	Fire	Water	Electric	Grass	Ice	Fighting	Poison	Ground	Flying	Psychic	Bug	Rock	Ghost	Dragon	Dark	Steel	Fairy
Normal	1x	1x	1x	1x	1x	1x	1x	1x	1x	1x	1x	1x	0.5x	0x	1x	1x	0.5x	1x
Fire	1x	0.5x	0.5x	1x	2x	2x	1x	1x	1x	1x	1x	2x	0.5x	1x	0.5x	1x	2x	1x
Water	1x	2x	0.5x	1x	0.5x	1x	1x	1x	2x	1x	1x	1x	2x	1x	0.5x	1x	1x	1x
Electric	1x	1x	2x	0.5x	0.5x	1x	1x	1x	0x	2x	1x	1x	1x	1x	0.5x	1x	1x	1x
Grass	1x	0.5x	2x	1x	0.5x	1x	1x	0.5x	2x	0.5x	1x	0.5x	2x	1x	0.5x	1x	0.5x	1x
Ice	1x	0.5x	0.5x	1x	2x	0.5x	1x	1x	2x	2x	1x	1x	1x	1x	2x	1x	0.5x	1x
Fighting	2x	1x	1x	1x	1x	2x	1x	0.5x	1x	0.5x	0.5x	0.5x	2x	0x	1x	2x	2x	0.5x
Poison	1x	1x	1x	1x	2x	1x	1x	0.5x	0.5x	1x	1x	1x	0.5x	0.5x	1x	1x	0x	2x
Ground	1x	2x	1x	2x	0.5x	1x	1x	2x	1x	0x	1x	0.5x	2x	1x	1x	1x	2x	1x
Flying	1x	1x	1x	0.5x	2x	1x	2x	1x	1x	1x	1x	2x	0.5x	1x	1x	1x	0.5x	1x
Psychic	1x	1x	1x	1x	1x	1x	2x	2x	1x	1x	0.5x	1x	1x	1x	1x	0x	0.5x	1x
Bug	1x	0.5x	1x	1x	2x	1x	0.5x	0.5x	1x	0.5x	2x	1x	1x	0.5x	1x	2x	0.5x	0.5x
Rock	1x	2x	1x	1x	1x	2x	0.5x	1x	0.5x	2x	1x	2x	1x	1x	1x	1x	0.5x	1x
Ghost	0x	1x	1x	1x	1x	1x	1x	1x	1x	1x	2x	1x	1x	2x	1x	0.5x	1x	1x
Dragon	1x	1x	1x	1x	1x	1x	1x	1x	1x	1x	1x	1x	1x	1x	2x	1x	0.5x	0x
Dark	1x	1x	1x	1x	1x	1x	0.5x	1x	1x	1x	2x	1x	1x	2x	1x	0.5x	1x	0.5x
Steel	1x	0.5x	0.5x	0.5x	1x	2x	1x	1x	1x	1x	1x	1x	2x	1x	1x	1x	0.5x	2x
Fairy	1x	0.5x	1x	1x	1x	1x	2x	0.5x	1x	1x	1x	1x	1x	1x	2x	2x	0.5x	1x

Figure 1: Pokemon Type Chart as of Generation 9

to rank each of them. Our first attempt at rankings will utilize a "king of the hill" (KOTH) style of ranking. For a game of KOTH among n players, one player is randomly selected to be the king and the other $n-1$ players are randomly chosen to challenge the king for his throne. This randomly selected player, sometimes called the challenger, will become the king if he defeats the king in battle. If the challenger loses, the king retains his throne.

Consider a KOTH game among all of the Pokemon types. One type is randomly chosen to start as the king, and then challengers will face the king in a 1v1 battle. The winner of this 1v1 battle is chosen in the following way: if the king has a higher damage multiplier when attacking the challenger, the king wins. If the challenger has a higher damage multiplier when attacking the king, the challenger wins. If each do equivalent damage to each other, the winner is randomly selected. For example, suppose the current king is the dragon type. If the fire type is selected as the challenger, dragon remains the king. If the ice type is chosen, ice becomes king. Lastly, if a type such as normal is selected, then we will flip a coin to determine the winner.

Now suppose that the KOTH game was played for an infinite number of rounds. Following from a previous argument, one type would not remain as the king the entire time because each type has its own poor match-ups. However, it stands to reason that over a large number of rounds, the "better" types would accumulate many rounds as the king whereas the less desirable types would spend most of their time as a challenger. Thus, one way to interpret the relative strength of a type is to compute its average time spent as king.

Mathematical Model

The above KOTH game and its analysis can be modeled very well by a Markov chain. In short, Markov chains can be used to analyze the behavior of a stochastic system over time. We will not go into the full details of these objects, but we can provide enough to get started.

A Markov chain consists of a set of states S, a set of actions A, and a probability transition matrix P. The set S denotes all the possible scenarios we could encounter. Lastly, the matrix P provides the probability that we reach a state i from state j. In our KOTH game, each state is uniquely described by which type is currently the king. Furthermore, the probability transition matrix P tells us the likelihood of each type becoming the next king. Let's consider a short example where there only exists three types: fire, water, and grass. In this scenario, our probability transition matrix would be:

$$P = \begin{pmatrix} 2/3 & 1/3 & 0 \\ 0 & 2/3 & 1/3 \\ 1/3 & 0 & 2/3 \end{pmatrix}.$$

The matrix is interpreted in the following way: the first row says that if our king is currently the fire type, then there is a 2/3 chance that the next king is fire, 1/3 that it is water, and 0 that it is grass. This is because grass will never win the 1v1 and water will always win the 1v1, but water only has a 1/3 chance at being randomly selected as the challenger (here, we assume that the challenger can be the same type as the king).

The probability transition matrix lets us do powerful simulations without having to explicitly play out the game. Suppose that we start with the fire type as the king. This is denoted using the initial vector $v = (1, 0, 0)^T$. We can quickly determine what the likelihood of the next king by applying P to our initial vector v. Indeed, $Pv = (2/3, 1/3, 0)^T$ which matches our previous argument that fire retains its throne with probability 2/3 and loses it to water with probability 1/3. To simulate more rounds of our KOTH game, we can just continually apply the matrix P to v.

Consider playing the KOTH game with fire, water, and grass for 100 rounds. Who should we expect the king to be after all the rounds are completed? One type is not necessarily favored over the other two, so it stands to reason that there should be an equal probability that each type is the king. Indeed, if we compute $P^{100}v = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})^T$ we see that each type has roughly an equal chance mathematically.

This is certainly helpful, but we are interested in the average time that each type spends as the king. Some more advanced Markov theory tells us that the stationary distribution (i.e. average time spent as king) of our Markov chain can be found by computing a solution to $P^T v = v$, i.e. computing an eigenvector with eigenvalue equal to 1. This can be done numerically in most standard programming languages. To continue with our example, the normalized eigenvector we are looking for turns out to be $(1/3, 1/3, 1/3)^T$. Unsurprisingly, this indicates that each type spends an equal amount of time as king and there is no separation between the three types.

Although our toy example turned out to be trivial, this approach provides us with a generalized way of ranking all the Pokemon types together. We first need to construct the probability transition matrix when all the types are considered. Then, we need to compute appropriate eigenvector of the transposed matrix. Doing this by hand would be next to impossible, so we'll utilize Python to compute these values.

The Dual Typing Case

Just as we argued that the fire/water/grass case could be extended to all of the types, we can also extend the 18 singular types into all 153 unique dual typings with a few modifications.

To compute the damage multiplier of one dual type attacking another, we will simply take the better of the two types of the attacker. For example, if a fire ghost type is attacking a grass dark type, the "best" attack multiplier is 2 when using a fire move. Similarly, if the grass dark type were attacking the fire ghost, the attack multiplier would also be 2 since dark hits fire ghost super effectively. Thus, if the king was the fire ghost type and the grass dark type was selected as the challenger, then there would be a 1/2 chance the next king would be fire ghost and a 1/2 chance the next king would be grass dark since both types hit the other for 2x damage.

Expanding our probability matrix to compute the transitions for all 153 types is even less manageable by hand, but again we can compute these eigenvectors using a programming language.

Results

With our methodology laid out for the Markov approach, we can now take a look at the results. To reiterate, we will be using computing the transition matrices and eigenvectors with the help of Python scripts which you can find here[1].

Single Type Rankings

The results of the single type rankings using the Markov approach are listed in Table 1. It should come as no surprise that steel ranks best - this will be a recurring theme among all of our results in this document. However, some of the middling pack may be surprising. Without going over each ranking individually, some surprising rankings are dragon as the fourth best typing according to this metric. This somewhat makes sense since dragon has a number of resistances and is resisted by only a few types. As a result, it has a high likelihood of winning random 1v1s. On the flip side, electric is considered by many to be a strong type. However, its lack of resistances makes it less likely to win standard 1v1s. Towards the bottom of the spectrum, it should come as no surprise to see types such as bug, psychic, and normal. Each of these types have very few resistances while also fail to hit a ton of types for super effective damage.

One final note should be made regarding the rock typing. Many consider rock to be a liability typing; however, our analysis suggests it is a very average type. The discrepancy is likely due to the distribution of moves that hit rock super effectively. Moves such as scald, earthquake, and close combat are extremely common and all

[1]https://github.com/trevorsquires/pokemon (underscore) type (underscore) rankings

Type Name	Probability
bug	0.035896123690678135
normal	0.03797228384469167
grass	0.040703899021189545
psychic	0.04247117547166929
poison	0.045699454638490546
ice	0.04622004057292578
dark	0.050876325514258885
flying	0.05223549303142918
fighting	0.05593167143033881
rock	0.05649000704842281
electric	0.058536119460233155
ground	0.05911370828781475
fairy	0.05926800222752867
fire	0.060218386414311036
dragon	0.06093011906822644
water	0.07200202284728366
ghost	0.0752748066477946
steel	0.0901603607827132

Table 1: Single Type Markov Rankings

Type 1 Name	Type 2 Name	Probability
grass	normal	0.0026794718009610563
dragon	grass	0.0031041426467650255
dragon	normal	0.003170539837838329
psychic	grass	0.0033230216839595823
bug	poison	0.0033380947973896125
poison	normal	0.003341029182233624
dark	grass	0.0033785834728864038
bug	normal	0.0033803821702914163
dragon	psychic	0.003415384413706846
bug	grass	0.0035327173708381097

Table 2: Dual Type Markov Rankings Bottom 10

do massive damage to rock types. If these moves were less widely distributed, this analysis suggests that the community would have a different opinion of the disrespected rock type.

Dual Type Rankings

With 153 dual types, it is not so easy to digest the full type rankings according to our Markov model. Instead, we list the bottom and top 10 types in Tables 2 and 3. Starting with the bottom types, we see grass appearing five different times in the bottom 10. This is unsurprising to some degree. Grass is notorious for its many weaknesses. It's redeeming quality is that it has useful resistances, even if there are not many of them. Unfortunately, in a setting that emphasizes winning many match-ups, grass does not do its dual types any favors. In fact, of all the typings it is paired with in the bottom 10, none help it against its weaknesses.

The other common types in the bottom 10 are normal, dragon, psychic, and bug. Of these, perhaps the most interesting is dragon. Although dragon was a great type in the single type setting, its inability to hit many types super effectively makes it a poor partner. Yes, it does provide many resistances, but this can also be achieved by having two functional types. Pairing it with normal, psychic, and bug makes it lose to the majority of all match-ups.

Moving on to the top 10, we see the common types of ground, steel, water, and fairy. These are generally accepted as the most versatile types so it makes sense to see them beside other types. Ground is an incredible offensive so pairing it with a type that hits its resistances or resists its weaknesses is quite strong. Perhaps the

Type 1 Name	Type 2 Name	Probability
fairy	water	0.009813191799263201
steel	water	0.010182521770215957
fairy	ground	0.01020482474359845
steel	ground	0.010525906447314406
steel	dragon	0.010805329592263506
flying	water	0.011449422029310315
steel	flying	0.0121545078945425
ground	fire	0.012835983124397394
flying	ground	0.013264634547844599
ground	water	0.013902775714238482

Table 3: Dual Type Markov Rankings Top 10

most important takeaway here is that just because a type is strong in a 1v1 match-up does not necessarily mean that it is the best typing partner.

3 Modeling as a Two-Player Zero-Sum Game

Continuing with the philosophy that a Pokemon's type is well represented by its ability to win a 1v1 matchup, we can take an alternative approach to determining a type's ranking. Instead of randomized approach to how we find ourselves in type matchups, suppose that we entered into a 1v1 typing tournament. We will be matched up against other participants and are only allowed to bring a single type. The rules of determining a winner in the 1v1 are the same as the previous section, but now our adversaries are not random type generators as in the Markov setting, but other players also trying to win the tournament.

Observe that this game falls into the category of a "Two Person, Zero Sum" game. Indeed, our goal is to maximize our payoff (the probability of winning the 1v1) while our adversary's goal is to minimize our payoff / maximize their payoff[2]. In particular, if we take actions that improve our odds of winning the matchup, this directly correlates to our opponent's odds decreasing since both must sum to 1. Given that our adversary is actively working against us, what strategy should we employ to give ourselves the best odds of winning the tournament?

Since we have to win multiple rounds to win the tournament and our opponents will inevitably learn a bit about our strategy of picking a type to bring, our goal is to pick a strategy that is the *least* exploitable. For example, suppose our strategy was to always bring the fire type. If our opponent caught wind of this technique, we would have a 0% chance of winning since they could exploit us by bringing a water type. If instead, we brought a normal type 50% of the time and a flying type the other 50% of the time, the opponent could never exploit our strategy with a single type. These strategies that involve a probability distribution over our possible actions are called mixed strategies.

3.1 Finding a Nash Equilibrium

How can we determine what our best mixed strategy is? We would like to choose our probabilities as ones that maximize our chance of winning regardless of what the adversary chooses to do. Let p_i represent the probability that we choose type i in a particular 1v1. If we knew the type that our opponent would choose, we would then be able to calculate our probability of winning. Let's go back to our three type setting for an example. Let p_1, p_2, p_3 be the probabilities that we pick fire, water, or grass respectively. Notice that since these variables represent probabilities, they must sum to 1 and all be greater than 0. What is our probability of winning if our adversary were to pick fire? By previous arguments, we would be guaranteed to win if the we pick water, guaranteed to lose if we pick grass, and a coin flip if we picked fire. Thus, our probability of winning is given by $0.5p_1 + p_2$. We

[2]This is actually a symmetric zero sum two-player game

can summarize the results for all three types as follows

$$\begin{aligned}P(\text{win if opponent pick fire}) &= 0.5p_1 + p_2\\ P(\text{win if opponent pick water}) &= 0.5p_2 + p_3\\ P(\text{win if opponent pick grass}) &= 0.5p_3 + p_1\end{aligned}$$

Recall that our objective is to choose our probabilities are least exploitable. Thus, we would like to maximize the *worst* case event. Here, the worst case win probability of winning is clearly the minimum of $0.5p_1 + p_2$, $0.5p_2 + p_3$, and $0.5p_3 + p_1$. We can then find our probabilities p_i by solving

$$\begin{aligned}&\max_p \min(E_1, E_2, E_3)\\ \text{s.t. } &\sum_i p_i = 1, p_i \geq 0\\ &E_1 = 0.5p_1 + p_2\\ &E_2 = 0.5p_2 + p_3\\ &E_3 = 0.5p_3 + p_1.\end{aligned}$$

However, the minimization in the objective can give us trouble analytically so we will instead introduce an auxiliary variable q and reformulate as

$$\begin{aligned}&\max_{p,z}\ z\\ \text{s.t. } &\sum_i p_i = 1, p_i \geq 0\\ &z \leq 0.5p_1 + p_2\\ &z \leq 0.5p_2 + p_3\\ &z \leq 0.5p_3 + p_1.\end{aligned}$$

Observe that in this second formulation, the value of z will take on the minimum of the three winning probabilities and is indeed the same as our first minimization. Again, it is not too hard to guess the optimal value of p in this simple toy example. The best choice of our probabilities is $p = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ and the best winning probability is $\frac{1}{2}$.

We have proven that, in our toy example, there is a strategy that *guarantees* we win 50% of the time. It is very natural to wonder if there exists a strategy that does even better. Unfortunately, if we assume that our adversaries are also well intending, they will also seek to maximize their worst case event. By an identical argument to the one we just proposed, our opponent can find a strategy that ensures they win at least 50% of the time which then limits us to a maximum of a 50% win rate at best.

The fact that our opponent can find a way to minimize our probability to exactly the amount that we can guarantee that we get is a result of the duality between the corresponding linear programs and more generally referred to as a Nash equilibrium. More generally, it can be shown that any two-player zero-sum game that allows mixed strategies will have at least one Nash equilibrium. Furthermore, since the game is symmetric, it does not matter whether we compute the optimal probabilities from the perspective of the minimizer or the maximizer, the result will be the same. We can use this idea to extend our toy example into a setting with all 18 different Pokemon types. We can then relate a high probability of being chosen in the Nash equilibrium as the relative strength of the Pokemon type.

3.2 Results

The application of this idea can be found in the Python scripts housed in the repo shared previously. The results are summarized in Table 4.

This is perhaps our most interesting ranking. Rather than a continuous stream of probabilities, the types are broken into tiers of identical probabilities. Furthermore, half the types are not recommended for use at all. This implies that any non zero usage of the type would have been better used by another type. To see why this is the case, take some time to evaluate how the "not recommended" types fare against the three highest ranked types in this list.

Type Name	Probability
normal	0
fire	0
fighting	0
poison	0
flying	0
psychic	0
bug	0
rock	0
dark	0
grass	0.05
ice	0.05
electric	0.1
ground	0.1
ghost	0.1
fairy	0.1
water	0.16
steel	0.16
dragon	0.16

Table 4: Two-Player Zero-Sum Rankings

Some ”not recommended” types are shocking at first. Fire is traditionally very good offensively and will appear highly on other lists. However, if we specifically focus on the types that have a non-zero probability in this list, we see that fire is only winning against grass, ice, and steel while weak to most everything else. In this sense, fire is not awarded for its strengths against the weakest types. The same can be said for the rest of the types in the ”not recommended” tier.

Among the non-zero probabilities, we see that the highest tier closely resembles what we've seen so far: water, steel and dragon are just very good in 1v1s. The middling tier is mostly familiar as well, with the exception of the electric type. It is clear that electric is ranked higher in this list in order to curb the effectiveness of water. Similarly, grass and ice are sometimes chosen just to serve as counters to water and dragon respectively.

4 Utilizing a Fixed Point Method

One of the obvious flaws with the two above approaches is that we reward types for being good in 1v1 match-ups. In particular, it is only important that types *win* the 1v1, but don't get rewarded for dominating a match-up. In addition, the Markov approach is a more holistic approach where we do not imprint any biases into the model. Consequently, we had some off putting types in the top of the single type rankings.

In this section, we will take a more user generated approach at these rankings. To do so, let's start over in our analysis of the types. First, we will split our analysis into two: computing an offensive **and** defensive rankings for each type. In the end, we can combine the the two into a single value. Note that this section will feature much more assumed Pokemon knowledge than the previous two and will be largely motivated by gameplay.

Type Strength Discussion

How does one determine the offensive value of a type? It stands to reason that a good offensive type is one that not only hits some types super effectively, but also one that hits the ”meta” defensive types, i.e. the types that are strong defensively. It doesn't make sense to heavily reward a type for hitting psychic and ice super effectively since those types would rarely see use as a defensive type. By the same token, if a type can only hit steel super effectively, that's still a valuable offensive type since one would expect to see steel frequently. Similarly, a defensive type is defined by both its resistances **and** *which* offensive types it resists.

Using this logic, we can compute a ”value” for a type's offensive ranking as follows. Suppose that we list each type as t_i for some i, i.e. t_1 =ghost, t_2 =normal, etc. Let $o(t)/d(t)$ denote the offensive/defensive value of type t and denote $m(t_i, t_j)$ to be the damage multiplier when type t_i attacks t_j. We assume that a Pokemon's offensive type is a linear combination of its damage multipliers and the other types' defensive values.

$$o(t_i) = \sum_j m(t_i, t_j)d(t_j).$$

Let's make things clearer with an example. Consider again the example where only fire/water/grass types exist. Then to compute how good the fire type is offensively, we would use

$$o(fire) = 2 \cdot d(grass) + 0.5 \cdot d(water) + 1 \cdot d(fire).$$

In this way, if grass was considered a strong defensive typing, then the fire type gets rewarded offensively for hitting it super effectively. Exactly how much the fire type is rewarded, however, is up for discussion. It seems natural to use the damage multiplier as a weight, but this can be debated. For example, an argument can be made that the most important aspect of an offensive move is the moves neutral coverage. We have seen with the introduction of the tinted lens ability that the ability to not have one's STAB moves resisted is very powerful. Thus, perhaps one would like to punish a type heavily for being resisted more than we would reward a type for hitting certain types super effectively. We will define the function $w(x)$ to map damage multipliers to "rewards". Thus, our new computation for an offensive type value is given by

$$o(t_i) = \sum_j w(m(t_i, t_j))d(t_j).$$

In the example of fire/water/grass, the function w satisfies $w(0.5) = 0.5, w(1) = 1, w(2) = 2$, but this need not be the case. Lastly, we can add one additional parameter to our formula for additional flexibility. As it stands, our value computation is a bilinear function of the reward due to type match-ups **and** which types it is hitting well. However, exactly how much this function should depend on the type its hitting vs how well its hitting is debatable. We can add an exponent p to help balance this. Thus, our final model for offensive value computation looks like

$$o(t_i) = \sum_j w(m(t_i, t_j))d(t_j)^p.$$

Below we will list all the offensive parameter choices that were made to generate the results later. If one would like to see the defensive parameters, that can be found in the github

$$p = 3, w(x) = \begin{cases} 0, & x = 0 \\ 0.25, & x = 0.25 \\ 0.5, & x = 0.5 \\ 1, & x = 1 \\ 1.25, & x = 2 \\ 1.5, & x = 1.5 \end{cases}.$$

Here, since $p = 3$, we are saying that the ability to perform well versus the best defensive types is very important. We will later choose $p = 1$ in the defensive setting. This implies that the ability to perform well against the best offensive types is less important. What's more important to a defensive type is all around resistances.

Our choice of w for our offensive formula is simply punishing a type for being resisted while barely rewarding it for hitting types well. We do still reward a type for being super effective against others, but less than we would punish for being resisted.

The Fixed Point Procedure

Now that we understand how to compute the offensive value for a type, we could do a similar process for computing the defensive values if we knew the offensive values of all the types. However, in both of these discussions, we assumed that we either know the offensive or defensive values to calculate the other - which is not true. To get started, we can assume that all types have an offensive and defensive value of 1, i.e. we are assuming that all types are equally strong both offensively and defensively. We then use the assumed defensive values to compute new offensive values, and then use our newly computed offensive values to compute new defensive values. We continue iterating through this process until our newly computed offensive/defensive values are unchanging.

On the mathematical side, this iterative process can be viewed as finding a fixed point of a dynamical system. There are certain conditions under which the fixed point is both guaranteed to exist and we are guaranteed to

Type Name	Offensive Value	Defensive Value	Overall Value
bug	70.29653363095277	87.94171660431054	71.67382801253244
grass	78.74852171222282	92.34876347921421	80.08830749115336
poison	71.98021341869024	101.12113429073966	82.73105752923684
ice	104.2567286359359	67.68728985391235	82.90232568116774
psychic	95.69519012035764	83.62005012452337	85.56634078472767
fighting	97.04973839016064	90.46995712831774	90.92874569369269
rock	110.1295766721497	80.73084310159386	94.6394756846749
normal	88.87868118732553	103.82036906455471	94.75204971323897
dragon	92.39475094619583	102.0300435939938	95.69116957321916
electric	104.3182079955357	103.35691386997917	104.50950346408162
flying	106.79703870113555	105.80961733135678	107.8511890628676
dark	111.83282190750292	105.3956740220947	111.04603730646042
ghost	111.92291876789197	106.87710818999119	112.08572240186446
fire	116.22746383655472	102.64861480544974	112.38297645958856
fairy	106.04333720733149	114.7224054928321	113.4938417080335
ground	125.72922774019598	98.64260475600967	116.92565798624703
water	115.31646856307223	113.9022269001252	119.10901689584102
steel	92.38258056678748	138.87466739100074	123.6227545513718

Table 5: Single Type Fixed Point Rankings

find it, but we won't go into that much detail. In summary, provided that the parameters are not extremely poorly chosen, we will be likely to converge to an offensive and defensive ranking within a few moments.

Once we have the offensive and defensive rankings, we can combine them into a singular type value. Exactly how this done is entirely up for discussion, but we have chosen to to use the following equation

$$overall = (\text{offensive}^n + \text{defensive}^n)^{1/n}$$

for some constant n. When $n = 1$, this is just the sum of the two values. When $n > 1$, however, the overall value of a type depends more on how good the higher of its two values is. For example, if $n = \infty$, then the overall value would just be the maximum of its offensive and defensive value. In other words, as n increases, the overall value is more correlated with how **specialized** a type is offensive or defensively. We chose to use $n = 2$ since we view specialization is somewhat important, but a good type must still have some balance to be good.

Lastly, as with the Markov approach, we are also able to use our single type values to compute dual type rankings in a similar manner. The process is nearly identical to the single type discussion, so we will not discuss this further.

Results

Our work to compute these rankings will be done via Python scripts that you can find on the aforementioned github link.

Single Type Rankings

Table 5 shows the computed offensive, defensive, and overall ranks for each single type. Unsurprisingly, when we add in some bias for how we value types, our results become much closer to our expectations. Steel remains the top type boasting a massive defensive value. It is multiple standard deviations above all other types due to the sheer number of resistances that it has. However, it is paired with the fourth-worst offensive rating. Because of this, its overall ranking is still number 1, but not exactly dominating.

Water and ground are the next two types. Water is an incredibly balanced type while ground is the top offensive typing. Neither of these two results are surprising. Moving towards the middle of the board, It is somewhat counter intuitive to see electric in the middle of the offensive pack. This would suggest that electric's coverage is not as good as the community believes it is, but rather is good when paired with other types (such as ice).

Towards the bottom of the table, we see the expected bug, grass, and ice. Poison is quite low considering it has a respectable defensive rating, but its offense is almost worst than bug. Speaking of bug, it is a terrible type.

First Type	Second Type	Offensive Value	Defensive Value	Overall Value
bug	grass	67.13777307891976	86.16825889002386	109.2357515382885
bug	poison	69.54586677980691	87.55568428243424	111.81513956669711
bug	ice	84.83325692185483	74.04944119087634	112.60551150210404
psychic	ice	93.08414465724172	63.63073310781832	112.75428232403729
ice	grass	85.77259835392759	73.95622588538903	113.25397112505534
rock	ice	94.20832033014324	66.49276063155773	115.3104281313382
dragon	ice	86.37478739974493	77.42970623184544	115.9998418339483
bug	psychic	87.59743676577007	76.17458697375164	116.08565212184811
rock	psychic	90.29475871571219	73.40736018644965	116.36917109385902
rock	bug	87.60111541410116	76.64660955059364	116.39870350393008

Table 6: Dual Type Fixed Point Rankings Bottom 10

First Type	Second Type	Offensive Value	Defensive Value	Overall Value
ground	water	113.4901436419419	117.36205145950369	163.2601109476892
steel	water	96.94140803091972	134.25204871730568	165.59362661591504
fairy	water	112.47699599770982	121.58291227729246	165.63055028133132
steel	electric	102.114400714083	130.6246170495279	165.80151209362396
steel	fighting	119.18212011980924	117.34227085446786	167.25306061634848
flying	ground	124.16580971196014	112.60360074158474	167.62076005494342
fairy	ground	127.05907352420826	110.14525934189655	168.15465001100566
steel	dark	107.22416241839473	130.81858770336217	169.1464569390342
steel	ghost	107.22416241839473	130.9855387656079	169.27561068282358
steel	ground	122.69812348042147	121.94996580194062	172.99313184260015

Table 7: Dual Type Fixed Point Rankings Top 10

In terms of overall rating, it is farther behind all the other types than steel is ahead of everyone else. That's not even considering its stealth rock weakness either.

Overall, however, the values match the communities views of each type quite well.

Dual Type Rankings

Tables 6 and 7 show the computed offensive, defensive, and overall ranks for the bottom 10 and top 10 dual types respectively. Starting with the bottom 10, our bug friend appears in five of the bottom 10 dual typings, sharing that distinction with ice. These types just don't bring enough offensively to justify their defensive ratings. Perhaps another interesting comment here is that none of the bottom 10 dual types have an immunity. Types that have an immunity are extremely valuable in a dual typing pair since they can completely cover for a weakness in the other type unlike a resistance which would simply be offset.

Moving on to Table 7, we see the power of immunities. Not only do all of the top 10 types have an immunity between them, five of them have multiple immunities. As expected, the top combination is steel-ground which is simply the pairing of the best offensive and defensive type. Furthermore, most of the top 10 is just steel paired with another strong type. Notable mentions must be given to the fairy-ground pairing which is the highest non-steel type pair and remarkably strong offensively. It is followed by flying-ground which is a more balanced pair, but recall that we *slightly* prefer specialized type pairings.

Lastly, perhaps the most surprising member in the top 10 is steel-dark. The model views this as one of the most defensive type pairings despite having a 4x weakness. It may be beneficial to slightly adjust the model to punish 4x typings in the future as this is a significant weakness to a type's defensive utility.

5 Conclusion

Overall, our type calculations have closely resembled that of the communities, but with a few distinctions. We believe that our approaches are a more holistic approach to ranking types and are generally free of biases. While there is still room to properly tune the parameters in the fixed point approach, these are ultimately subjective decisions to be made. We also believe that our work can serve as a starting point for determining which tera-types will become popular since these decisions should be made without the natural biases present.